Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/KR05/000776

International filing date: 18 March 2005 (18.03.2005)

Document type: Certified copy of priority document

Document details: Country/Office: KR
Number: 10-2004-0018279
Filing date: 18 March 2004 (18.03.2004)

Date of receipt at the International Bureau: 19 April 2005 (19.04.2005)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)

PCT/KR 2005 / 0 0 0 1 7 6
RO/KR 1 8. 0 3. 2005

대 한 민 국 특 허 청
KOREAN INTELLECTUAL PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office.

출 원 번 호 : 10-2004-0018279
Application Number

출 원 년 월 일 : 2004년 03월 18일
Date of Application MAR 18, 2004

출 원 인 : 학교법인고려중앙학원
Applicant(s) KOREA CHUNGANG EDUCATIONAL FOUNDATION

2004 년 12 월 27 일

특 허 청 장
COMMISSIONER

【서지사항】

【서류명】	특허출원서
【권리구분】	특허
【수신처】	특허청장
【참조번호】	0010
【제출일자】	2004.03.18
【국제특허분류】	G06F
【발명의 명칭】	버퍼 오버플로우 공격들을 감지하고 복구하는 방법 및 그 장치
【발명의 영문명칭】	Method for sensing and recovery against buffer overflow attacks and apparatus thereof
【출원인】	
【명칭】	학교법인 고려중앙학원
【출원인코드】	2-1995-276862-2
【대리인】	
【성명】	이영필
【대리인코드】	9-1998-000334-6
【포괄위임등록번호】	2002-018861-4
【대리인】	
【성명】	이해영
【대리인코드】	9-1999-000227-4
【포괄위임등록번호】	2002-018862-1
【발명자】	
【성명의 국문표기】	최린
【성명의 영문표기】	CHOI,Lynn
【주민등록번호】	640210-1026028
【우편번호】	480-020
【주소】	경기도 의정부시 호원동 신일유토빌아파트 101동 1303호
【국적】	KR
【발명자】	
【성명의 국문표기】	신용
【성명의 영문표기】	SHIN,Yong
【주민등록번호】	781015-1065435

1020040018279

출력 일자: 2004/12/27

【우편번호】	135-231
【주소】	서울특별시 강남구 일원1동 662-8
【국적】	KR
【심사청구】	청구
【취지】	특허법 제42조의 규정에 의한 출원, 특허법 제60조의 규정에 의한 출원심사 를 청구합니다. 대리인 이영필 (인) 대리인 이해영 (인)
【수수료】	
【기본출원료】	42 면 38,000 원
【가산출원료】	0 면 0 원
【우선권주장료】	0 건 0 원
【심사청구료】	18 항 685,000 원
【합계】	723,000 원
【감면사유】	학교
【감면후 수수료】	361,500 원
【첨부서류】	1. 요약서·명세서(도면)_1통

【요약서】

【요약】

본 발명은 버퍼 오버플로우 공격들을 감지하고 복구하는 방법 및 그 장치를 개시한다.

본 발명에 의하면, 버퍼 오버플로우 공격으로부터 프로세서의 동작 상태를 복구하는 방법에 있어서, 버퍼 오버플로우 공격이 있을 수 있는 쓰기 동작들을 쓰기 동작들이 기록될 원래의 목적지 대신에 다른 장소에 저장하면서 쓰기 동작들 중에서 버퍼 오버플로우 공격이 있는가를 감지하고, 버퍼 오버플로우 공격이 있는 것으로 감지되지 않는 경우에는 소정의 간격으로 저장된 내용을 원래의 쓰기 동작들이 기록될 목적지에 저장하며, 버퍼 오버플로우 공격이 있는 것으로 감지되는 경우 공격으로 인한 안전하지 않은 쓰기들은 버리고, 버퍼 오버플로우 공격이 있는 것으로 감지되면 그 이후의 안전하지 않은 쓰기들을 저장하지 않고 무시하여, 효과적으로 컴퓨터에 발생될 수 있는 버퍼 오버플로우 공격을 감지하고, 공격을 당한 경우에도 피해를 최소화하면서 공격 이전의 상태로 복구/복귀할 수 있으며, 구현방법에 따라 컴퓨터 시스템의 성능의 저하는 최소화하면서 효과적으로 시스템을 방어할 수 있게 하므로 그 결과 컴퓨터와 인터넷을 사용하는 환경을 크게 개선할 수 있다.

【대표도】

도 3

【명세서】

【발명의 명칭】

버퍼 오버플로우 공격들을 감지하고 복구하는 방법 및 그 장치 {Method for sensing and recovery against buffer overflow attacks and apparatus thereof}

【도면의 간단한 설명】

도 1은 스택 스매싱 공격이 어떻게 x86계열 프로세서의 복귀 주소를 변경할 수 있는가를 보여준다.

도 2는 버퍼 오버플로우 공격에 의해 만들어진 제어 및 데이터 변조의 종류들을 분류한 것이다.

도 3은 본 발명에 따라 버퍼 오버플로우 공격을 감지하는 방법의 흐름의 한 예를 도시한 것이다.

도 4는 도 3의 방법을 실시하기 위한 버퍼 오버플로우 공격을 감지 장치의 구성을 블록으로 도시한 것이다.

도 5는 본 발명에 따른 버퍼 오버플로우 공격을 감지하는 방법의 다른 예의 흐름을 도시한 것이다.

도 6은 도 5의 방법을 실시하기 위한 버퍼 오버플로우 공격을 감지하는 장치의 구성을 블록으로 도시한 것이다.

도 7은 본 발명에 따른 복귀 주소 포인터 스택의 구조를 보여준다.

도 8은 변조 복구 버퍼를 이용하여 본 발명에 따라 버퍼 오버플로우 공격으로부터 복구하는 방법의 흐름을 도시한 것이다.

도 9는 도 8의 방법을 실시하기 위한 버퍼 오버플로우 공격으로부터 복구 장치의 구성을 블록으로 도시한 것이다.

도 10은 FIFO로 구성된 본 발명에 따른 변조 복구 버퍼의 구조를 보여준다.

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<11> 본 발명은 컴퓨터에 관한 것으로서, 컴퓨터 동작을 방해하는 버퍼 오버플로우 공격들을 감지하고 복구하는 하는 방법 및 그 장치에 관한 것이다.

<12> 버퍼 오버플로우(buffer overflow) 공격은 Code Red와 SQL Slammer 웜 (Worm) 등과 같은 최근의 웜의 발발에서 알 수 있는 것과 같이 가장 강력하고 치명적인 형태의 악성 코드 공격이다.

<13> 본 발명은 그와 같은 악성 코드를 이용한 공격들을 탐지하고 복구할 수 있는 방법을 제공한다.

<14> 버퍼 오버플로우 공격은 일반적으로 시스템에 데이터를 파괴하고 또 프로그램 실행 흐름을 변경시키는 것과 같이 비정상적인 증상들을 야기하는데, 이러한 증상들은 프로그램 실행 중에 명령어 참조와 데이터 참조의 안전성을 점검함으로써 하드웨어가 간단하게 감지할 수 있다. 이러한 명령어 참조와 데이터 참조의 안전 점검 장치와 더불어 보안의 수준을 증대시키기 위한 방안으로 변조 복구 버퍼(Corruption Recovery Buffer, CRB)라고 불리는 좀 더 강도 높은 기술을 본 발명은 제안한다. 안전 점검 장치들과 결합되어 CRB는 버퍼 오버플로우 공격으로 인해

야기되는 의심스러운 메모리 기록을 임시로 저장함으로써 공격이 감지되는 경우 메모리의 상태를 공격 이전의 상태로 복구시킬 수 있다.

<15> 버퍼 오버플로우에 대한 취약성과 이를 이용한 악성 코드 공격은 인터넷/컴퓨터 보안 문제 중에서 가장 치명적인 형태의 보안 문제이다. 첫 번째 이유는 버퍼가 오버플로우되는 것은 버퍼 근방의 데이터를 덮어쓰는 것 뿐 아니라 프로그램에 대한 제어를 가로채 악의의 목적을 가진 임의의 코드를 실행할 수 있기 때문이다. 두 번째로 악성 코드는 사람의 조작이 없이도 스스로 복제, 전파되기 때문에 악성 코드 공격의 모든 형태들 중에서도 가장 빠른 전파 속도를 가지기 때문이다. 세 번째로, 버퍼 오버플로우 공격은 1988년 최초의 웜 바이러스가 이러한 버퍼 오버플로우 공격의 형태로 시작한 것을 비롯하여 가장 오랫동안, 또 앞으로도 가장 지속적으로 공격이 예상되기 때문이다. 마지막으로, 버퍼오버플로우 공격은 악성 코드 공격 중에서 가장 자주 발생하는 공격 형태이기도 하다.

<16> 다양한 소프트웨어적인 해결 방안이 운영 체제의 수정, 컴파일러 도구들과 패치들, 디버깅 툴 및 동적 라이브러리 등의 형태로 제안되었으나 그들은 근본적으로 레거시(legacy) 응용 프로그램에는 속수무책이거나 혹은 심각한 성능 장애 요인을 포함하기도 한다. 여전히 대응책으로 가장 많이 사용되는 것은 취약점이 발견된 부분의 소스 코드를 수정 후 다시 컴파일 하여 패치 및 수정 사항들을 제작한 후 이를 해당 프로그램 사용자들로 하여금 직접 내려 받게 하는 것이다. 그러나 이는 취약한 소스가 공개적으로 알려진 후에 특정된 제품의 특정한 취약성을 더 알릴뿐이며 근본적인 대책은 아니다.

【발명이 이루고자 하는 기술적 과제】

<17> 본 발명이 이루고자 하는 기술적인 과제는, 상기의 문제점들을 해결하기 위해, 컴퓨터에 있어서 발생되는 악성 코드를 이용한 버퍼 오버플로우 공격들을 감지하고 복구할 수 있는 방법 및 그 장치를 제공하는데 있다.

【발명의 구성 및 작용】

<18> 상기 기술적 과제를 해결하기 위한 본 발명에 의한, 버퍼 오버플로우 공격 감지 방법의 일 태양은, (a) 프로세서 복귀 명령어를 가져올 때에 복귀 명령어가 가리키는 주소를 감지하는 단계; (b) 상기 감지된 주소가 프로세서의 스택 영역에 존재하는 가를 판단하는 단계; 및 (c) 상기 감지된 주소가 스택 영역에 존재하는 경우 상기 복귀 명령은 잘못되었으며 버퍼 오버플로우 공격이 있는 것으로 판단하는 단계;를 포함하는 것을 특징으로 한다.

<19> 상기 다른 기술적 과제를 해결하기 위한 본 발명에 의한, 버퍼 오버플로우 공격 감지 장치의 일 태양은, 프로세서 복귀 명령어를 가져올 때에 복귀 명령어가 가리키는 주소를 감지하는 주소감지부; 상기 주소감지부에서 감지한 주소가 프로세서의 스택 영역에 존재하는 가를 판단하는 확인부; 및 상기 확인부에서 상기 주소감지부에서 감지한 주소가 스택 영역에 존재하는 것으로 판단하는 경우 상기 복귀 명령은 잘못되었으며 버퍼 오버플로우 공격이 있는 것으로 판단하는 공격판단부;를 포함하는 것을 특징으로 한다.

<20> 상기 기술적 과제를 해결하기 위한 본 발명에 의한, 버퍼 오버플로우 공격 감지 방법의 다른 태양은, (a) 소정의 저장 명령을 수행한 후에 복귀되는 주소를 감지하는 단계; (b) 프로세서의 스택 영역 내의 연속적인 저장 명령이 상기 복귀되는 주소를 수정하는 가를 판단하는 단계; 및 (c) 상기 (b) 단계에서 연속적인 저장 명령이 상기 복귀되는 주소를 수정하는 것으로

판단되면 버퍼 오버플로우 공격이 있는 것으로 판단하는 단계;를 포함하는 것을 특징으로 한다.

<21> 상기 다른 기술적 과제를 해결하기 위한 본 발명에 의한, 버퍼 오버플로우 공격 감지 장치의 다른 태양은, 소정의 저장 명령을 수행한 후에 복귀되는 주소를 감지하는 주소감지부; 프로세서의 스택 영역 내의 연속적인 저장 명령이 상기 주소감지부에 의해 복귀되는 주소를 수정하는 가를 판단하는 주소수정판단부; 및 상기 주소수정판단부에서 연속적인 저장 명령이 상기 복귀되는 주소를 수정하는 것으로 판단하면 버퍼 오버플로우 공격이 있는 것으로 판단하는 공격판단부;를 포함하는 것을 특징으로 한다.

<22> 상기 기술적 과제를 해결하기 위한 본 발명에 의한, 버퍼 오버플로우 공격으로부터 복구 방법은, 버퍼 오버플로우 공격으로부터 프로세서의 동작 상태를 복구하는 방법에 있어서, (a) 버퍼 오버플로우 공격이 있을 수 있는 쓰기 동작들을 쓰기 동작들이 기록될 원래의 목적지 대신에 소정의 다른 장소에 저장하면서 상기 쓰기 동작들 중에서 버퍼 오버플로우 공격이 있는가를 감지하는 단계; (b) 버퍼 오버플로우 공격이 있는 것으로 감지되지 않는 경우에는 소정의 간격으로 상기 저장된 내용을 원래의 쓰기 동작들이 기록될 목적지에 저장하며, 버퍼 오버플로우 공격이 있는 것으로 감지되는 경우 공격으로 인한 안전하지 않은 쓰기들은 버리는 단계; 및 (c) 버퍼 오버플로우 공격이 있는 것으로 감지되면 그 이후의 안전하지 않은 쓰기들을 저장하지 않고 무시하는 단계;를 포함하는 것을 특징으로 한다.

<23> 상기 다른 기술적 과제를 해결하기 위한 본 발명에 의한, 버퍼 오버플로우 공격으로부터 복구 장치는, 버퍼 오버플로우 공격으로부터 프로세서의 동작 상태를 복구하는 장치에 있어서, 버퍼 오버플로우 공격이 있을 수 있는 쓰기 동작들을 쓰기 동작들이 기록될 원래의 메모리 영역 대신에 소정의 다른 저장 수단에 저장하는 저장부; 상기 쓰기 동작들 중에서 버퍼 오버플

로우 공격이 있는가를 감지하는 공격감지부; 버퍼 오버플로우 공격이 있는 것으로 감지되지 않는 경우에는 소정의 간격으로 상기 소정의 다른 저장수단에 저장된 내용을 원래의 쓰기 동작들이 기록될 메모리 영역에 저장하며, 버퍼 오버플로우 공격이 있는 것으로 감지되는 경우 공격으로 인한 안전하지 않은 쓰기들은 버리는 저장관리부; 및 버퍼 오버플로우 공격이 있는 것으로 감지되면 그 이후의 안전하지 않은 쓰기들을 상기 소정의 다른 저장 수단에 저장하지 않고 무시하는 기록관리부;를 포함하는 것을 특징으로 한다.

<24> 이하에서 첨부된 도면을 참조하여 본 발명의 바람직한 일 실시예를 상세히 설명한다.

<25> 버퍼 오버플로우 공격은 하드웨어에 의해 간단하게 감지될 수 있는 이상한 증상이나 흔적을 나타내는 것이 보통이다. 예를 들면 스택 스매싱이라고 불리는 버퍼 오버플로우 공격의 가장 흔한 형태는 프로세스의 로컬 스택 프레임의 복귀 주소를 변경하며 이는 보통 프로그램 수행 중에는 가능하지 않다. 마찬가지로 악성 코드들은 종종 프로그램 주소 공간의 스택이나 데이터 영역으로부터 명령어를 가져오려 하는데, 이 역시 보통의 프로그램 수행에서는 잘 일어나지 않는 동작이다.

<26> 이러한 비정상적인 증상들을 검출하기 위해 본 발명은 명령어와 데이터 참조들의 주소 영역을 검사하여 그 안전성을 검증하는 안전 보호 장치(safety guard)라 불리는 마이크로구조 기술을 도입한다. 또한 의심스러운 쓰기들을 저장하여 공격이 감지될 경우 시스템을 공격 이전의 상태로 되돌릴 수 있는 변조 복구 버퍼(Corruption recovery buffer; CRB)라는 더욱 강도 높은 보안 기술을 제시한다.

<27> 버퍼 오버플로우 공격은 오버플로우된 변수 근방의 변조되는 데이터에 의해 시작된다. 프로세스 주소 공간의 프로그램 텍스트 영역은 쓰기가 금지 되어 있기 때문에 스택, 힙 혹은

정적 데이터와 같은 데이터 영역들만이 변조된다. 도 1은 스택 스매싱 공격이 어떻게 x86계열 프로세서의 복귀 주소를 변경할 수 있는가를 보여준다.

<28> x86 프로세서에서 EBP는 스택의 바닥, ESP는 스택의 위를 가리키며 EBP와 ESP가 가리키는 영역은 한 프로시져에 해당하는 스택 프레임(100)이다. 스택 프레임(100)에는 지역 변수(110), 인자(120), 복귀 주소(130), 이전 프레임 포인터 (EBP, 140)가 저장되어 있다. 여기서 지역 변수(110) 또는 인자(120)가 오버플로우 공격의 대상이 되는 버퍼 변수이며 이러한 버퍼 공격은 참조번호 100 스택 프레임의 아래에서 위로 스택에 있는 인근 데이터를 덮어쓰게 되며 결국은 복귀 주소(130)를 변경시키게 된다. 이 변경된 복귀주소(150)는 일반적으로 외부로부터 입력된 악성 코드의 시작 주소를 가리키며 현재 공격이 발생한 함수가 복귀될 때 변경된 복귀 주소(150)로의 복귀, 즉 악성코드를 시작하여 악성코드가 실행되는 것이다.

<29> strcpy와 같이 취약점이 있는 함수가 지역 변수에 입력을 읽어들일 때 외부 입력은 함수의 스택 프레임 내에서 해당 변수 뿐만 아니라 인근 영역의 데이타로 오버플로우되며 이는 결과적으로 복귀 주소를 변조시킬 수 있다.

<30> 도 2는 버퍼 오버플로우 공격에 의해 만들어진 제어 및 데이터 변조의 종류들을 분류한 것이다. 언제 그리고 어디에서 데이터가 변조되었는가에 따라 프로그램 실행 도중 에러 발생 여부가 정해진다. 변조된 데이터 항목이 그 이후로 전혀 참조되지 않는다면 데이터 변조는 프로그램 실행에 영향을 미치지 않을 것이다. 이하에서는 이런 경우를 "사용하지 않는 데이터 변조" 라고 한다. 만일 같은 위치에서 더 이른 시점에 동일한 공격이 가해지면 그 데이터는 참조될 것이고 심각한 문제를 일으킬 수 있다. 그러므로 데이터 변조의 효과는 그 위치만이 아니라 공격의 시점에도 영향을 받는다. 변조된 데이터 항목이 나중에 참조되는 경우를 이하에서는 "활동성 데이터 변조"라고 언급할 것이다.

<31> 활동성 데이터 변조는 다음과 같이 3가지 종류로 분류될 수 있다. 첫 번째로 변조된 데이터가 리턴 (return) 명령 혹은 간접 분기(indirect branch) 명령과 같은 분기 명령어의 목적 주소(target)로 사용되는 경우 이를 "목적 주소 변조"라 부른다. 이런 경우 변경된 분기 목적 주소로부터 명령어를 가져올 경우 프로그램의 실행 흐름이 변경된다. 대부분의 버퍼 오버플로우 공격은 스택 프래임의 복귀 주소 혹은 스택 또는 힙 영역에서의 함수 포인터들을 목표로 하여 이와 같은 제어 데이터의 변조를 통하여 악성 코드 공격을 시도한다.

<32> 데이터 변조의 두 번째 종류는 공격이 분기 명령의 조건을 변경할 때에 발생한다. 이것 역시 복귀 주소나 함수 포인터를 변경하지 않더라도 프로그램 제어 흐름에 대한 변경을 유발할 수 있다. 이하에서는 이와 같은 것을 "분기 조건 변조"라고 할 것이다. 예를 들어

<33> if (a>0) then call func_a(a) else func_b(a);

<34> 과 같은 코드에서 변수 a의 값에 따라 다른 함수가 인출될 수 있다. 즉 변수 a의 값이 변조된다면 정상적으로 인출될 함수가 아닌 다른 함수를 인출함으로써 제어 흐름을 변경하는 것이다.

<35> 이하에서는 활동성 데이터 변조의 다른 모든 경우를 "순수 데이터 변조"라고 할 것이다. 이 경우 제어의 흐름을 직접적으로 바꾸지는 않지만 이후 프로그램에 의해 참조될 활동성 데이터를 변경한다. 이런 경우 정상적인 프로그램의 실행처럼 보이지만 잘못된 결과나 상태를 초래할 수 있다. 덧붙여 데이터 변조는 다른 데이터 위치들로 전파될 수 있으며, 그 결과 분기 목표 변조 혹은 분기 조건 변조로 이어질 수 있다.

<36> 상기의 활동성 데이터 변조의 3가지 경우 모두 비정상적인 실행 또는 종료를 일으킬 수 있다. 변조된 데이터 항목을 참조하는 것은 잘못된 피연산자들에 의해 발생되는 에러와 관련된

데이터 참조 예외 혹은 실행으로 이어지기 쉽다. 예를 들면 변조된 데이터 주소로부터 로드하는 것은 TLB 미스(miss), 페이지 미스 혹은 접근 위반으로 이어질 수 있다. 비록 변조된 데이터 항목이 성공적으로 참조된다 하더라도 이후의 연산 과정에서 오버플로우나 플로팅 포인트 예외 등을 일으킬 수 있다.

<37> 변조된 데이터 항목이 조건 분기의 분기 조건으로 혹은 간접 분기의 분기 목표로 사용되는 경우 실행 제어 흐름을 변경시킨다. 특히 잘못된 목표 주소로부터 명령어를 가져오는 경우를 "제어 변조"라고 부른다. 이것은 버퍼 오버플로우 공격에 의해 발생한 잘못된 제어 흐름을 의미한다. 이는 분기 목표가 외부에서 들어온 웜 코드에 대한 포인터로 변경될 경우 악성 코드 실행으로 이어질 수 있다. 이하에서는 이와 같은 제어 변조를 "외부 코드 제어 변조"라고 할 것이다. 악성 코드는 스스로 복제되어 다른 취약한 호스트들로 전파되기 때문에 버퍼 오버플로우 공격의 가장 심각한 결과라 할 수 있다. 또한 이것은 버퍼 오버플로우 공격의 가장 흔한 형태이기도 하다. 제어 변조의 또 다른 형태는 변경된 분기 목표가 텍스트 영역의 정상적인 코드를 가리킬 때이다. 이것을 "내부 코드 제어 변조"라고 부를 것이다.

<38> 이하 본 발명에 따라 버퍼 오버플로우 공격 감지를 하는 안전 점검의 흐름을 설명한다.

<39> 공격을 당하는 시스템은 데이터와 명령어 참조 시 비정상적인 증상을 보인다. 예를 들면 스택 스매싱 공격은 스택 내부의 지역 변수들 뿐 아니라 복귀 주소까지도 덮어쓰게 된다. 게다가 종종 현재의 스택 프레임 밖의 스택 영역으로 자신에게 동반된 악의의 코드를 복사하는데, 이 어느 경우도 보통의 프로그램 실행 중에는 가능한 것이 아니다. 또한 공격을 당한 프로그램은 스택 영역으로부터 명령어들을 가져오게 된다. 리눅스의 시그널 핸들러 혹은 gcc의 trampolines 함수들과 같은 아주 드문 경우들 이외에는 스택으로부터 명령어를 가져오는 것은

흔한 일이 아니다. 비정상적인 명령어 또는 비정상적인 데이터 참조는 참조되고 있는 주소에 대한 안전 검사를 통해 실행 도중 쉽게 탐지할 수 있다.

<40> 도 3은 본 발명에 따라 버퍼 오버플로우 공격을 감지하는 방법의 흐름의 한 예를 도시한 것이다.

<41> 이 방법은 프로세서 복귀 명령어를 가져올 때에 복귀 명령어가 가리키는 주소를 감지하고(300 단계), 감지된 주소가 프로세서의 스택 영역에 존재하는가를 판단하며(310 단계), 감지된 주소가 스택 영역에 존재하는 경우 상기 복귀 명령은 잘못되었으며 버퍼 오버플로우 공격이 있는 것으로 판단한다(320 단계).

<42> 도 4는 도 3의 방법을 실시하기 위한 버퍼 오버플로우 공격을 감지 장치의 구성을 블록으로 도시한 것이다.

<43> 이 장치는, 프로세서 복귀 명령어를 가져올 때에 복귀 명령어가 가리키는 주소를 감지하는 주소감지부(400), 주소감지부(400)에서 감지한 주소가 프로세서의 스택 영역에 존재하는 가를 판단하는 확인부(410) 및 확인부(410)에서 주소감지부(400)에서 감지한 주소가 스택 영역에 존재하는 것으로 판단하는 경우 상기 복귀 명령은 잘못되었으며 버퍼 오버플로우 공격이 있는 것으로 판단하는 공격판단부(420)를 포함한다.

<44> 이 경우 공격판단부(420)는 320 단계에서 잘못된 명령으로 판단된 복귀 명령은 실행하지 않고 버리는 것이 바람직하다.

<45> 이하 본 발명이 CPU와 같은 프로세서에 포함되어 구현된 경우를 예를 들어 상세하게 설명한다. 이는 프로세서가 어떻게 버퍼 오버플로우 공격에 의한 데이터 혹은 제어 변조에 대항하여 시스템을 방어할 수 있는가를 보여준다.

<46> 프로세서에서 명령어를 가져올 때에 도 4의 장치는 결국 다음과 같은 동작을 하는 것이다.

<47> 복귀 명령이 가리키는 주소가 스택 영역에 존재할 때, 복귀 명령은 잘못된 것으로 판단한다.

<48> 만일 프로그램 카운터가 스택 영역 내의 어느 한 지점을 가리킨다면 해당 명령어는 안전하지 않은 것으로 판단되어 버려지게 된다. 하지만 이러한 방법은 gcc trampolines 함수들 또는 리눅스 시그널 핸들러와 같이 스택 영역으로부터 명령어를 가져와야 하는 예외적인 경우 문제가 될 수 있다.

<49> 버퍼 오버플로우를 발생시키는 악성 코드들은 많은 경우 복귀 주소를 덮어쓰기 때문에, 스택 영역으로 제어를 바꾸는 명령어는 복귀 명령어가 된다. 하지만, trampolines 또는 리눅스 시그날 핸들러의 경우는 복귀 명령어가 아닌 호출 (call) 명령어에 의해 불려진다. 그러므로 복귀 명령어의 목표 주소가 스택 영역을 가리키는지를 검사함으로써, trampolines 함수들로 인한 정상적인 명령어들과 악성 코드 공격으로 인한 비정상적인 명령어들을 구분할 수 있다. 명령어를 실행하는 동안의 이러한 종류의 안전 검사를 안전 점검이라고 하며 복귀 명령어에 대한 특정한 경우의 참조 안전 검사를 명령어 참조 안전 점검이라고 부른다.

<50> 이것은 간단한 범위 체크이며 본 발명을 구현하는 경우 하드웨어적인 비용 또는 성능 면에서의 손실 없이도 하드웨어로 구현될 수 있다.

<51> 외부로부터 들어온 악성 코드는 프로그램의 주소 공간 중 대부분의 경우 스택 영역에만 존재할 수 있기 때문에 명령어 참조 안전 점검은 버퍼 오버플로우 공격에 의해 만들어진 외부 코드 제어 변조를 효과적으로 제거할 수 있다. 그러나 이는 텍스트 영역으로부터 명령어들을

가져오는 내부 코드 제어 변조는 막기 힘든 경우가 발생할 수 있다. 또한 이것은 분기 조건이 아닌 분기 목표 주소만을 검사하므로 분기 조건 변조에 의해 발생되는 제어 변조 역시 막기 힘들 수 있다.

<52> 공격의 초기 단계에서 공격을 감지하여 막는 것이 언제나 더 나은 방법이다. 그러므로 데이터 변조 단계에서 시스템을 방어하는 것이 제어 변조 후에 시스템을 방어하는 것보다 시스템에 가해지는 피해를 줄일 수 있다. 명령어 참조 안전 보호 외에 다음과 같은 또 다른 안전 보호 방안이 데이터 변조 단계에서 도 5와 같이 제공될 수 있다.

<53> 도 5는 본 발명에 따른 버퍼 오버플로우 공격을 감지하는 방법의 다른 예의 흐름을 도시한 것이다.

<54> 이 방법은, 소정의 저장 명령을 수행한 후에 복귀되는 주소를 감지하고(500 단계), 프로세서의 스택 영역 내의 연속적인 저장 명령이 상기 복귀되는 주소를 수정하는가를 판단하며(510 단계), 510 단계에서 연속적인 저장 명령이 상기 복귀되는 주소를 수정하는 것으로 판단되면 버퍼 오버플로우 공격이 있는 것으로 판단한다(520 단계).

<55> 도 6은 도 5의 방법을 실시하기 위한 버퍼 오버플로우 공격을 감지하는 장치의 구성을 블록으로 도시한 것이다.

<56> 이 장치는 소정의 저장 명령을 수행한 후에 복귀되는 주소를 감지하는 주소감지부(600), 프로세서의 스택 영역 내의 연속적인 저장 명령이 주소감지부(600)에 의해 복귀되는 주소를 수정하는 가를 판단하는 주소수정판단부(610) 및 주소수정판단부(610)에서 연속적인 저장 명령이 상기 복귀되는 주소를 수정하는 것으로 판단하면 버퍼 오버플로우 공격이 있는 것으로 판단하는 공격판단부(620)를 포함한다.

<57> 도 6의 장치는 결과적으로 다음과 같은 동작을 하는 것이다.

<58> 스택 영역 내의 연속적인 저장 명령이 복귀 주소를 수정할 때, 이는 비정상적인 것으로 판단한다.

<59> 복귀 주소가 저장된 곳의 위치에 대해 연속적인 저장 명령어들의 주소 영역을 검사함으로써 복귀 주소 혹은 프레임 포인터의 변조로부터 시스템을 보호할 수 있다. 이러한 안전 보호를 데이터 참조 안전 점검이라고 한다. 복귀 주소가 수정되는 것을 방지함으로써 스택 스매싱 공격에 의해 제어 변조가 일어나는 것을 막을 수 있다. 이 안전 보호를 사용하기 위해, 복귀 주소들이 저장되는 곳의 주소들을 알고 있어야 한다. 이를 위해 이 경우 주소감지부(600)는 참조번호 500 단계에서 소정의 저장 명령을 수행한 수에 복귀되는 주소를 소정의 스택에 저장하며, 주소수정판단부(610)는 상기 연속적인 저장 명령의 저장 범위가 상기 스택에 저장된 주소와 상충되는가를 판단하여 상기 복귀되는 주소가 침범되어 수정되는 것으로 판단하는 것이 바람직하다.

<60> 이에 대한 설명을 위해 본 발명에서는 복귀 주소 포인터 스택(Return Address Pointer Stack, 이하 RAPS라 함)의 개념을 사용한다.

<61> 도 7은 본 발명에 따른 RAPS의 구조를 보여준다. 복귀 주소가 스택에 저장될 때마다, 저장되는 곳의 주소가 RAPS 에 푸쉬(push) 된다. 복귀 주소가 스택으로부터 읽혀질 때, RAPS에서 팝(pop) 연산이 일어난다. x86 프로세서에서 setjmp, longjmp 와 같은 경우에는 아래 수식과 같이 여러번의 팝 (Pop) 연산이 한번에 일어나야 한다.

<62> while (RAPS(top) < ESP) pop RAPS;

<63> 두 개의 연속적인 저장 연산이 스택 내의 낮은 주소에서 높은 주소 방향으로 이어지는 연속적인 영역에 일어날 때, RAPS 는 시작 지점과 끝 지점을 기억하는 두 개의 포인터(각각 StartPtr과 EndPtr)를 이용해 이 범위를 기억하게 된다. 만일 연속적인 저장 범위가 RAPS 의 Top에 저장된 주소를 침범하게 되면 (즉 연속적인 저장 범위와 RAPS에 저장된 주소 사이에 충돌이 발생하면) 데이터 참조 안전 점검 장치는 안전 위반을 알리게 된다.

<64> 대부분의 버퍼 오버플로우 공격은 가장 바깥쪽의 스택 프레임(stack frame) 부근의 복귀 주소를 변조시킨다. 만일 오버플로우된 변수가 현재 스택 프레임 내의 지역 변수인 경우, 바로 이전 스택 프레임 내의 복귀 주소가 악성 코드에 의해 변조된다. 하지만 오버플로우된 변수가 포인터에 의한 호출 (call by reference)로 전달된 인자인 경우, 실제 버퍼 오버플로우는 이전 스택 프레임에서 발생하게 되고, 그 하나 이전의 스택 프레임 내의 복귀 주소가 변조되게 된다. 즉, 오버플로우가 일어나는 변수가 참조에 의한 호출 방식으로 전달될 경우, 어떠한 스택 프레임에서도 버퍼 오버플로우 공격이 발생할 수 있다. 그러므로 RAPS 내의 모든 값들에 대해 연속적인 저장의 범위가 검사되어야 한다. 하지만 대부분의 공격들은 가장 최근의 스택 프레임 부근의 복귀 주소를 목표로 하므로, 몇 개의 최근 복귀 주소만 검사하는 것도 충분히 좋은 방법이 될 수 있다.

<65> 상기와 같이 동작되는 데이터 참조 안전 점검 장치가 구현되는 경우 하드웨어 비용 면에서는 명령어 참조 안전 점검에 비해 더 비싸지만, 제어 및 데이터 변조에 대해서는 더 높은 수준의 방어를 제공하게 된다.

<66> 버퍼 오버플로우 공격으로부터의 원래 상태 복구

<67> 프로세서와 같은 소자에 구현될 수 있는 상기에 설명된 본 발명에 따른 안전 점검 장치에 의해 위험한 메모리 참조들을 탐지하게 되면 몇 가지 대응 방법을 선택할 수 있다.

<68> 첫 번째, 공격을 당하는 프로세스를 끝낼 수 있다. 그러나 이것은 공격자가 프로세스를 끝내는데 성공하는 것이므로 서비스 거부 상태 (Denial of Service Condition)를 발생시킬 수 있다.

<69> 두 번째 선택은 현재 공격당하는 함수의 실시를 끝내고 그 제어 흐름을 강제로 호출자에게로 돌리는 것이다. 이것을 강제 복귀(compulsory return)라고 한다. 이것은 본 발명에 따른 장치가 복귀 주소의 위치를 추적하여 강제로 PC 값을 바꿀 수 있기 때문에 가능하다.

<70> 세 번째는 모든 안전하지 않은 저장 연산들을 무시하고, 원래의 실행 흐름은 변화시키지 않는 방법이다. 이것을 조용한 복구(calm recovery) 라고 한다. 이 방법은 원래의 실행 흐름은 바꾸지 않지만, 지역 변수들에 대한 데이터 변조들로 인해 비정상적인 실행을 유발할 수 있다.

<71> 마지막 방법은 공격 이전의 상태로 복구시킨 후 호출자에게로 복귀시키는 것이다. 그러나 이것은 이전 또는 새로운 구조적인 상태를 저장하기 위한 하드웨어적인 저장 수단 혹은 버퍼를 필요로 한다.

<72> 본 발명에서는 버퍼 오버플로우 공격들을 감지할 때의 복구 혹은 복귀 기술로 상기의 방법을 복합하여 사용할 수 있다. 이하 상기의 각 복구 혹은 복귀 방법에 대해 설명한다.

<73> 강제 복귀(compulsory return)

<74> 간단한 하드웨어 해결책으로 공격당한 함수를 끝내고 그 제어의 흐름이 호출자에게 복귀되도록 강제로 변경한다. 이는 스택에 저장된 복귀 주소의 값으로 프로그램 카운터를 갱신하고 현재의 스택 프레임을 팝업(pop up)하여 구현될 수 있다. 많은 경우 공격당한 함수의 목적은 단지 외부의 입력을 로컬 변수로 복사하는 것 뿐이다. 그러므로 오버플로우가 발생한 변수는 악의의 코드만을 포함하고 있으므로 대부분의 경우 입력을 버리는 것이 안전하다.

<75> 조용한 복구

<76> 공격으로 인해 발생하는 모든 안전하지 않은 쓰기들을 단순히 무시하는 방법이다. 데이터 참조 안전 점검 장치는 복귀 주소를 보호하기 때문에, 공격은 제어를 가로챌 수 없으며, 공격을 당한 프로세스는 공격 이후에 정상적인 실행을 계속하게 될 것이다. 프로세서는 조용히 안전하지 않은 쓰기들만을 무시하고 공격당한 프로세스의 원래의 제어 흐름을 바꾸지 않으므로 이것을 조용한 복구(calm recovery) 라고 부른다. 그러므로 이 방법은 강제 복귀 방법에 비해 덜 강제적이다. 하지만 이 방법은 데이터 변조에 취약하며, 데이터 참조 안전 점검 장치가 안전 위반을 알리기 이전에 스택 프레임 내의 지역 변수들이 이미 변조되므로 비정상적인 실행을 유발할 수 있다.

<77> 변조 복구 버퍼(CRB)

<78> 더 적극적인 방법은 공격이 감지되었을 때 원래의 버퍼 상태를 복구하는 것이다. 이를 위해, 변조 복구 버퍼(CRB)라고 불리는 특별한 하드웨어 버퍼에 의심스러운 쓰기들을 저장한다. 데이터 참조 안전 보호 장치 위반이 발생했을 때, 프로세서는 CRB내의 모든 안전하지 않은 값들을 버리고, 조용히 그 이후의 안전하지 않은 쓰기들을 무시한다. 그리고 그 이후 이어지는 모든 연산들을 조용한 복구 (calm recovery) 에서와 같이 수행하게 된다. 이렇게 함으로써, 메모리의 입력 버퍼는 원래 상태를 유지하게 되며, 어떠한 종류의 스택 영역에 대한 데이터 변조도 피할 수 있게 된다.

<79> 도 8은 CRB를 이용하여 본 발명에 따라 버퍼 오버플로우 공격으로부터 복구하는 방법의 흐름을 도시한 것이다.

<80> 프로세서의 동작 상태를 복구하는 이 방법은, 버퍼 오버플로우 공격이 있을 수 있는 쓰기 동작들을 쓰기 동작들이 기록될 원래의 목적지 대신에 소정의 다른 장소에 저장하면서(800 단계), 상기 쓰기 동작들 중에서 버퍼 오버플로우 공격이 있는가를 감지한다(810 단계). 그리고 버퍼 오버플로우 공격이 있는 것으로 감지되지 않는 경우에는 소정의 간격으로 상기 저장된 내용을 원래의 쓰기 동작들이 기록될 목적지에 저장하며, 버퍼 오버플로우 공격이 있는 것으로 감지되는 경우 공격으로 인한 안전하지 않은 쓰기들은 버리며(820 단계), 버퍼 오버플로우 공격이 있는 것으로 감지되면 그 이후의 안전하지 않은 쓰기들을 저장하지 않고 무시한다(830 단계).

<81> 도 9는 도 8의 방법을 실시하기 위한 버퍼 오버플로우 공격으로부터 복구 장치의 구성을 블록으로 도시한 것이다.

<82> 버퍼 오버플로우 공격으로부터 프로세서의 동작 상태를 복구하는 이 장치는, 버퍼 오버플로우 공격이 있을 수 있는 쓰기 동작들을 쓰기 동작들이 기록될 원래의 메모리 영역(910) 대신에 소정의 다른 저장 수단(920)에 저장하는 저장부(900), 상기 쓰기 동작들 중에서 버퍼 오버플로우 공격이 있는가를 감지하는 공격감지부(930), 버퍼 오버플로우 공격이 있는 것으로 감지되지 않는 경우에는 소정의 간격으로 소정의 다른 저장수단(920)에 저장된 내용을 원래의 쓰기 동작들이 기록될 메모리 영역(910)에 저장하며, 버퍼 오버플로우 공격이 있는 것으로 감지되는 경우 공격으로 인한 안전하지 않은 쓰기들은 버리는 저장관리부(940), 버퍼 오버플로우 공격이 있는 것으로 감지되면 그 이후의 안전하지 않은 쓰기들을 소정의 다른 저장 수단(920)에 저장하지 않고 무시하는 기록관리부(950)를 포함한다.

<83> 그리고 이 장치는 상기 쓰기 동작들이 기록될 원래의 메모리(910) 영역으로의 읽기 동작을 관리하는 인출관리부(960)를 더 포함하여, 인출관리부(960)는 상기 쓰기 동작들의 원래의 메모리(910) 영역으로 읽기 동작이 있는 경우 원래의 메모리(910) 영역과 저장부(900)에 의해 저장되는 소정의 다른 저장 수단(920)에 동시에 접근하여, 읽기 동작의 목적지 주소가 소정의 다른 저장 수단(920)에 있으나 아직 원래의 메모리(910) 영역에 저장되지 않은 경우에는 소정의 다른 저장 수단(920)으로부터 장소로부터 상기 읽기 동작이 요구하는 데이터를 공급하는 것이 바람직하며, 저장부(900)가 저장하는 소정의 다른 저장 수단(920)은 선입선출 방식의 FIFO 소자임이 바람직하다.

<84> 또한 저장관리부(940)는 버퍼 오버플로우 공격이 감지되는 경우 공격이 발생하지 이전에 저장된 쓰기들도 원래의 쓰기 동작들이 기록될 메모리 영역(910)에 저장하는 것이 바람직하다.

<85> 그 외에도 저장부(900)는 상기 쓰기 동작들이 기록될 원래의 메모리 영역(910)의 연속적인 영역에서 두 번째의 계속적인 쓰기들이 있는 경우에는 상기 쓰기들을 소정의 다른 저장 수

단(920)에 저장하며, 그렇지 않은 경우에는 쓰기 동작들이 기록될 원래의 메모리 영역(910)에 기록하는 것이 바람직하다. 이 동작을 위해 저장관리부(940)의 제어를 받는다.

<86> 그리고 저장관리부(940)가 FIFO 소자(920)에 저장된 내용을 원래의 쓰기 동작들이 기록될 메모리 영역에 저장하는 소정의 간격은 FIFO 소자(920)의 용량으로 정해질 수 있는 것이 바람직하다.

<87> 도 10은 FIFO로 구성된 본 발명에 따른 CRB의 구조를 보여준다. 이하의 설명은 필요시에는 도 9의 도면과 동시에 참조번호를 인용한다.

<88> CRB(1000)는 잠재적인 버퍼 오버플로우 공격의 결과인 의심스러운 쓰기들을 저장하는데 사용된다. 이 의심스러운 쓰기들이 안전한 것으로 판명된 이후 이 값들은 메모리(1010)에 전달된다. 그러므로 메모리(1010)는 공격 이전의 버퍼의 원래 상태를 항상 유지하게 된다. 메모리(1010)는 일반의 시스템 메모리 또는 CPU 내부 혹은 CPU에 연계되어 있는 캐시 메모리가 될 수 있으며, 어느 경우에도 본 발명의 요지는 유지되면서 구현될 수 있다.

<89> 버퍼 오버플로우 공격은 메모리의 연속적인 영역에 계속적인 쓰기들을 실행하는 것을 동반한다. 단일 쓰기는 메모리의 버퍼를 넘치게 하지 않기 때문에 본 발명에서는 바람직하게 메모리(1010)의 연속적인 영역에 두 번째의 계속적인 쓰기를 의심스러운 쓰기의 시작으로 간주한다. 그러므로 두 개의 계속적인 저장 연산만이 아니라 메모리의 연속적인 영역들도 의심스러운 공격의 성립을 위한 필수적 조건이다. 이하에서는 이와 같은 연속적인 메모리 영역의 두 번째 기록을 CRB 트리거라고 언급할 것이다. 저장관리부(940, 1020)가 CRB 트리거에 따라 CRB에 데이터를 기록하고 이 기록된 데이터를 원래 기록되어야 할 메모리(1010)에 저장되게 한다.

<90> 저장관리부(1020)는 CRB 트리거가 발생하지 않는 경우에는 버퍼 오버플로우 공격이 아닌 것이므로 CRB(1000)에 기록할 필요없이 바로 목적지 메모리(1010)에 기록한다. 만일 CRB 트리거가 발생한 경우라면 이것은 의심스러운 쓰기일 가능성이 있으므로 모든 쓰기는 저장관리부(1020)에 의해 원래 기록되어야 할 메모리(1010)가 아닌 CRB(1000) 특히 테일 부분에 기록된다.

<91> CRB 트리거가 저장관리부(1020)에 의해 감지되면 다음의 계속적인 쓰기들이 CRB 트리거 조건이 충족되지 않을 때까지 저장부(900)에 의해 메모리(1010) 대신 CRB(1000)에 쓰여진다. 만일 이 계속적인 쓰기들이 실행되는 중에 공격감지부(930)에 의해 버퍼 오버플로우 공격 즉, 데이터 참조 안전 점검 장치 위반이 발생하지 않는다면 CRB(1000)에 저장된 그 계속적인 기록들은 안전한 것으로 여겨져서 메모리(1010)로 전달된다.

<92> 데이터 참조 안전 점검 장치에 대한 위반이 발생하면 CRB(1000)에 저장된 값들은 안전하지 않으며 기록관리부(950)에 의해 그 이후의 기록들은 무시될 것이다. 또한 조용한 복구(calm recovery) 에서와 마찬가지로 본 발명이 적용된 프로세서는 모든 안전하지 않은 쓰기들을 무시한 후, 정상 실행을 계속할 것이다. 이 버퍼 오버플로우 공격 동안에 그 메모리는 그대로 있는 것이므로 데이터도 제어도 변조되지 않는다.

<93> 프로세서와 같은 장치에 본 발명에 따른 CRB(1000) 혹은 도 9의 구성을 추가하는 것은 읽기 연산을 복잡하게 한다. 최신의 데이터가 그 메모리에 있지 않을 수 있으므로 읽기는 CRB와 메모리를 동시에 접근해야 한다. 이에 대한 기능을 인출관리부(960, 1030)가 지원한다.

<94> 읽기 연산의 주소가 메모리(1010)에만 있는 경우(Miss)에는 인출관리부(1030)는 메모리(1010)로부터 그 데이터가 공급되도록 한다. 그리고 만일 읽기 연산의 주소가 CRB(1000)에 쓰

여진 데이터도 있으며 메모리(1010)에도 있는 경우에는 CRB(1000), 메모리(1010)의 어느 쪽에서 데이터를 공급해도 될 것이다.

<95> 만일 읽기 연산의 주소가 CRB(1000)에 쓰여진 데이터에 있으나 아직 메모리(1010)로 전달되지 않았다면(Hit) CRB(1000)가 그 데이터를 공급해야 한다. 이것은 CRB(1000)에 의해 유지되는 시작과 끝 주소를 읽기 연산의 주소와 비교하여 간단하게 검사될 수 있다. 도 10에 도시된 바와 같이 헤드(Head)와 테일(Tail)은 각각 아직 메모리(1010)로 전달하지 않은 CRB(1000)에서 가장 오래된 기록과 가장 최신의 기록을 가리킨다. CRB(1000)는 계속적인 쓰기들의 가장 나중의 실행만을 보관하고 있으므로 CRB의 이전 값들은 다른 CRB 트리거 조건이 만족되기 전에 메모리로 플러시(flush)되어야 한다. 그러므로 CRB 트리거가 감지되면 지연(stall)이 발생될 것이다.

<96> 지연은 CRB(1000)를 구성하는 FIFO 소자와 같은 저장 소자의 제한된 크기 때문에도 발생한다. 만일 CRB(1000)가 가득 차게 되면 의심스러운 기록들은 CRB(1000)의 새로운 데이터의 일부가 메모리(1010)로 플러시될 때까지 지연되어야 한다.

<97> 위의 경우와 유사하게 지연의 길이는 CRB(1000)의 크기에 달려있다. CRB가 작을수록 지연은 더 길게 더 자주 발생한다.

<98> 본 발명을 적용한 시스템의 성능을 시험하기 위해 SPEC2000 CPU 벤치마크, CPU2000 벤치마크의 다양한 어플리케이션들을 이용한 시험과 SimpleScalar 2.0 툴을 사용하여 기본 캐시/TLB를 가지며 4개의 무질서한 파이프라인을 가지면서 본 발명에 따른 CRB를 포함하는 프로세서 모델을 사용하여 각 어플리케이션마다 50개의 독립적인 시뮬레이션을 한 결과 1KB의 CRB는 2% 이하의 성능 감소로 모든 제어 및 데이터 변조를 제거하는데 충분한 것을 알 수 있다.

<99> 따라서 본 발명을 적용한 프로세서는 성능의 면에서는 저하되는 것이 거의 없으면서 버퍼 오버플로우 공격에 대해 아주 효율적인 수단을 제공하는 것을 알 수 있으며, 이는 본 발명의 효과를 충분히 입증한다고 할 수 있다.

<100> 상기에 설명된 본 발명에 따른 방법과 장치는 다양하게 구현될 수 있다. 예를 들면 ASIC 혹은 FPGA와 같은 소자를 사용해서 별도의 칩으로 구현하여 CPU와 같은 프로세서와 연동하여 사용되도록 구현될 수도 있고, 혹은 프로세서의 내부의 일 구성부로 구현되어 버퍼 오버플로우 프로세서를 구현하기 위해 사용될 수도 있다. 이와 같은 다양한 구현이 가능하다는 것은 본 발명이 속한 기술 분야의 통상의 지식을 가진 자는 용이하게 알 수 있는 것이다.

<101> 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자는 본 발명이 본 발명의 본질적인 특성에서 벗어나지 않는 범위에서 변형된 형태로 구현될 수 있음을 이해할 수 있을 것이다. 그러므로 본 개시된 실시예들은 한정적인 관점이 아니라 설명적인 관점에서 고려되어야 한다. 상기의 설명에 포함된 예들은 본 발명에 대한 이해를 위해 도입된 것이며, 이 예들은 본 발명의 사상과 범위를 한정하지 않는다. 상기의 예들 외에도 본 발명에 따른 다양한 실시 태양이 가능하다는 것은, 본 발명이 속한 기술 분야에 통상의 지식을 가진 사람에게는 자명할 것이다. 본 발명의 범위는 전술한 설명이 아니라 청구범위에 나타나 있으며, 그와 동등한 범위 내에 있는 모든 차이점은 본 발명에 포함된 것으로 해석되어야 할 것이다.

<102> 또한 본 발명에 따른 상기의 각 단계는 일반적인 프로그래밍 기법을 이용하여 소프트웨어적으로 또는 하드웨어적으로 다양하게 구현할 수 있다는 것은 이 분야에 통상의 기술을 가진 자라면 용이하게 알 수 있는 것이다.

<103> 그리고 본 발명의 일부 단계들은, 또한, 컴퓨터로 읽을 수 있는 기록매체에 컴퓨터가 읽을 수 있는 코드로서 구현하는 것이 가능하다. 컴퓨터가 읽을 수 있는 기록매체는 컴퓨터 시스템에 의하여 읽혀질 수 있는 데이터가 저장되는 모든 종류의 기록장치를 포함한다.

【발명의 효과】

<104> 본 발명에 의하면, 버퍼 오버플로우 공격으로부터 프로세서의 동작 상태를 복구하는 방법에 있어서, 버퍼 오버플로우 공격이 있을 수 있는 쓰기 동작들을 쓰기 동작들이 기록될 원래의 목적지 대신에 다른 장소에 저장하면서 쓰기 동작들 중에서 버퍼 오버플로우 공격이 있는가를 감지하고, 버퍼 오버플로우 공격이 있는 것으로 감지되지 않는 경우에는 소정의 간격으로 저장된 내용을 원래의 쓰기 동작들이 기록될 목적지에 저장하며, 버퍼 오버플로우 공격이 있는 것으로 감지되는 경우 공격으로 인한 안전하지 않은 쓰기들은 버리고, 버퍼 오버플로우 공격이 있는 것으로 감지되면 그 이후의 안전하지 않은 쓰기들을 저장하지 않고 무시하여, 효과적으로 컴퓨터에 발생될 수 있는 버퍼 오버플로우 공격을 감지하고, 공격을 당한 경우에도 피해를 최소화하면서 공격 이전의 상태로 복구/복귀할 수 있으며, 구현방법에 따라 컴퓨터 시스템의 성능의 저하는 최소화하면서 효과적으로 시스템을 방어할 수 있게 하므로 그 결과 컴퓨터와 인터넷을 사용하는 환경을 크게 개선할 수 있다.

【특허청구범위】

【청구항 1】

버퍼 오버플로우 공격을 감지하는 방법에 있어서,

(a) 프로세서 복귀 명령어를 가져올 때에 복귀 명령어가 가리키는 주소를 감지하는 단계;

(b) 상기 감지된 주소가 프로세서의 스택 영역에 존재하는 가를 판단하는 단계; 및

(c) 상기 감지된 주소가 스택 영역에 존재하는 경우 상기 복귀 명령은 잘못되었으며 버퍼 오버플로우 공격이 있는 것으로 판단하는 단계;를 포함하는 것을 특징으로 하는 버퍼 오버플로우 공격 감지 방법.

【청구항 2】

제1항에 있어서,

상기 (c) 단계에서 잘못된 명령으로 판단된 복귀 명령은 실행하지 않고 버리는 것을 특징으로 하는 버퍼 오버플로우 공격 감지 방법.

【청구항 3】

버퍼 오버플로우 공격을 감지하는 장치에 있어서,

프로세서 복귀 명령어를 가져올 때에 복귀 명령어가 가리키는 주소를 감지하는 주소감지부;

상기 주소감지부에서 감지한 주소가 프로세서의 스택 영역에 존재하는 가를 판단하는 확인부; 및

상기 확인부에서 상기 주소감지부에서 감지한 주소가 스택 영역에 존재하는 것으로 판단하는 경우 상기 복귀 명령은 잘못되었으며 버퍼 오버플로우 공격이 있는 것으로 판단하는 공격판단부;를 포함하는 것을 특징으로 하는 버퍼 오버플로우 공격 감지 장치.

【청구항 4】

제3항에 있어서,

상기 공격판단부는 버퍼 오버플로우 공격으로 잘못된 명령으로 판단된 복귀 명령은 더 이상 실행되지 않도록 무시하는 것을 특징으로 하는 버퍼 오버플로우 공격 감지 장치.

【청구항 5】

버퍼 오버플로우 공격을 감지하는 방법에 있어서,

(a) 소정의 저장 명령을 수행한 후에 복귀되는 주소를 감지하는 단계;

(b) 프로세서의 스택 영역 내의 연속적인 저장 명령이 상기 복귀되는 주소를 수정하는가를 판단하는 단계; 및

(c) 상기 (b) 단계에서 연속적인 저장 명령이 상기 복귀되는 주소를 수정하는 것으로 판단되면 버퍼 오버플로우 공격이 있는 것으로 판단하는 단계;를 포함하는 것을 특징으로 하는 버퍼 오버플로우 공격 감지 방법.

【청구항 6】

제5항에 있어서,

상기 (a) 단계에서 상기 소정의 저장 명령을 수행한 수에 복귀되는 주소를 소정의 스택에 저장하며,

상기 (b) 단계에서 상기 연속적인 저장 명령의 저장 범위가 상기 스택에 저장된 주소와 상충되는가를 판단하여 상기 복귀되는 주소가 침범되어 수정되는 것으로 판단하는 것을 특징으로 하는 버퍼 오버플로우 공격 감지 방법.

【청구항 7】

버퍼 오버플로우 공격을 감지하는 장치에 있어서,

소정의 저장 명령을 수행한 후에 복귀되는 주소를 감지하는 주소감지부;

프로세서의 스택 영역 내의 연속적인 저장 명령이 상기 주소감지부에 의해 복귀되는 주소를 수정하는 가를 판단하는 주소수정판단부; 및

상기 주소수정판단부에서 연속적인 저장 명령이 상기 복귀되는 주소를 수정하는 것으로 판단하면 버퍼 오버플로우 공격이 있는 것으로 판단하는 공격판단부;를 포함하는 것을 특징으로 하는 버퍼 오버플로우 공격 감지 장치.

【청구항 8】

제7항에 있어서,

상기 주소감지부는 상기 소정의 저장 명령을 수행한 수에 복귀되는 주소를 소정의 스택에 저장하며,

상기 주소수정판단부는 상기 연속적인 저장 명령의 저장 범위가 상기 스택에 저장된 주소와 상충되는가를 판단하여 상기 복귀되는 주소가 침범되어 수정되는 것으로 판단하는 것을 특징으로 하는 버퍼 오버플로우 공격 감지 장치.

【청구항 9】

버퍼 오버플로우 공격으로부터 프로세서의 동작 상태를 복구하는 방법에 있어서,

(a) 버퍼 오버플로우 공격이 있을 수 있는 쓰기 동작들을 쓰기 동작들이 기록될 원래의 목적지 대신에 소정의 다른 장소에 저장하면서 상기 쓰기 동작들 중에서 버퍼 오버플로우 공격이 있는가를 감지하는 단계;

(b) 버퍼 오버플로우 공격이 있는 것으로 감지되지 않는 경우에는 소정의 간격으로 상기 저장된 내용을 원래의 쓰기 동작들이 기록될 목적지에 저장하며, 버퍼 오버플로우 공격이 있는 것으로 감지되는 경우 공격으로 인한 안전하지 않은 쓰기들은 버리는 단계; 및

(c) 버퍼 오버플로우 공격이 있는 것으로 감지되면 그 이후의 안전하지 않은 쓰기들을 저장하지 않고 무시하는 단계;를 포함하는 것을 특징으로 하는 버퍼 오버플로우 공격으로부터 복구 방법.

【청구항 10】

제9항에 있어서,

상기 (b) 단계에서 버퍼 오버플로우 공격이 감지되는 경우 공격이 발생하지 이전에 저장된 쓰기들도 원래의 쓰기 동작들이 기록될 목적지에 저장하는 것을 특징으로 하는 버퍼 오버플로우 공격으로부터 복구 방법.

【청구항 11】

제9항에 있어서,

상기 (a) 단계에서 상기 쓰기 동작들이 기록될 원래의 목적지의 연속적인 메모리 영역에서 두 번째의 계속적인 쓰기들이 있는 경우에는 상기 쓰기들을 소정의 다른 장소에 저장하며, 그렇지 않은 경우에는 쓰기 동작들이 기록될 원래의 목적지에 기록하는 것을 특징으로 하는 버퍼 오버플로우 공격으로부터 복구 방법.

【청구항 12】

제9항에 있어서,

상기 쓰기 동작들의 원래의 목적지로 읽기 동작을 통해 접근하는 경우 상기 원래의 목적지와 상기 쓰기 동작들이 저장되는 장소로 동시에 접근하며,

읽기 동작의 주소가 상기 쓰기 동작들이 저장되는 장소에 있으나 아직 원래의 목적지에 저장되지 않은 경우에는 상기 쓰기 동작들이 저장되는 장소로부터 상기 읽기 동작이 요구하는 데이터를 공급하는 것을 특징으로 하는 버퍼 오버플로우 공격으로부터 복구 방법.

【청구항 13】

버퍼 오버플로우 공격으로부터 프로세서의 동작 상태를 복구하는 장치에 있어서,

버퍼 오버플로우 공격이 있을 수 있는 쓰기 동작들을 쓰기 동작들이 기록될 원래의 메모리 영역 대신에 소정의 다른 저장 수단에 저장하는 저장부;

상기 쓰기 동작들 중에서 버퍼 오버플로우 공격이 있는가를 감지하는 공격감지부;

버퍼 오버플로우 공격이 있는 것으로 감지되지 않는 경우에는 소정의 간격으로 상기 소정의 다른 저장수단에 저장된 내용을 원래의 쓰기 동작들이 기록될 메모리 영역에 저장하며, 버퍼 오버플로우 공격이 있는 것으로 감지되는 경우 공격으로 인한 안전하지 않은 쓰기들은 버리는 저장관리부; 및

버퍼 오버플로우 공격이 있는 것으로 감지되면 그 이후의 안전하지 않은 쓰기들을 상기 소정의 다른 저장 수단에 저장하지 않고 무시하는 기록관리부;를 포함하는 것을 특징으로 하는 버퍼 오버플로우 공격으로부터 복구 장치.

【청구항 14】

제13항에 있어서,

상기 저장관리부는 버퍼 오버플로우 공격이 감지되는 경우 공격이 발생하지 이전에 저장된 쓰기들도 원래의 쓰기 동작들이 기록될 메모리 영역에 저장하는 것을 특징으로 하는 버퍼 오버플로우 공격으로부터 복구 장치.

【청구항 15】

제13항에 있어서,

상기 저장부는 상기 쓰기 동작들이 기록될 원래의 메모리 영역의 연속적인 영역에서 두 번째의 계속적인 쓰기들이 있는 경우에는 상기 쓰기들을 소정의 다른 저장 수단에 저장하며, 그렇지 않은 경우에는 쓰기 동작들이 기록될 원래의 메모리 영역에 기록하는 것을 특징으로 하는 버퍼 오버플로우 공격으로부터 복구 장치.

【청구항 16】

제13항에 있어서,

상기 쓰기 동작들이 기록될 원래의 메모리 영역으로의 읽기 동작을 관리하는 인출관리부를 더 포함하여,

상기 인출관리부는 상기 쓰기 동작들의 원래의 메모리 영역으로 읽기 동작이 있는 경우 상기 원래의 메모리 영역과 상기 저장부에 의해 저장되는 소정의 다른 저장 수단에 동시에 접근하여, 읽기 동작의 목적지 주소가 상기 소정의 다른 저장 수단에 있으나 아직 원래의 메모리 영역에 저장되지 않은 경우에는 상기 소정의 다른 저장 수단으로부터 장소로부터 상기 읽기

동작이 요구하는 데이터를 공급하는 것을 특징으로 하는 버퍼 오버플로우 공격으로부터 복구 장치.

【청구항 17】

제13항 내지 제16항의 어느 한 항에 있어서,

상기 저장부가 저장하는 소정의 다른 저장 수단은 선입선출 방식의 FIFO 소자임을 특징으로 하는 버퍼 오버플로우 공격으로부터 복구 장치.

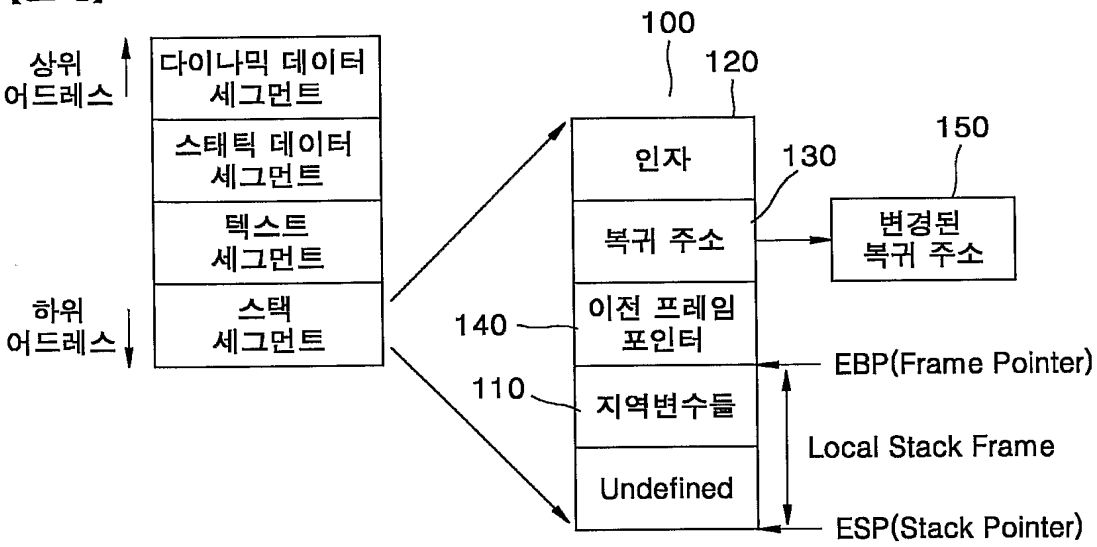
【청구항 18】

제17항에 있어서,

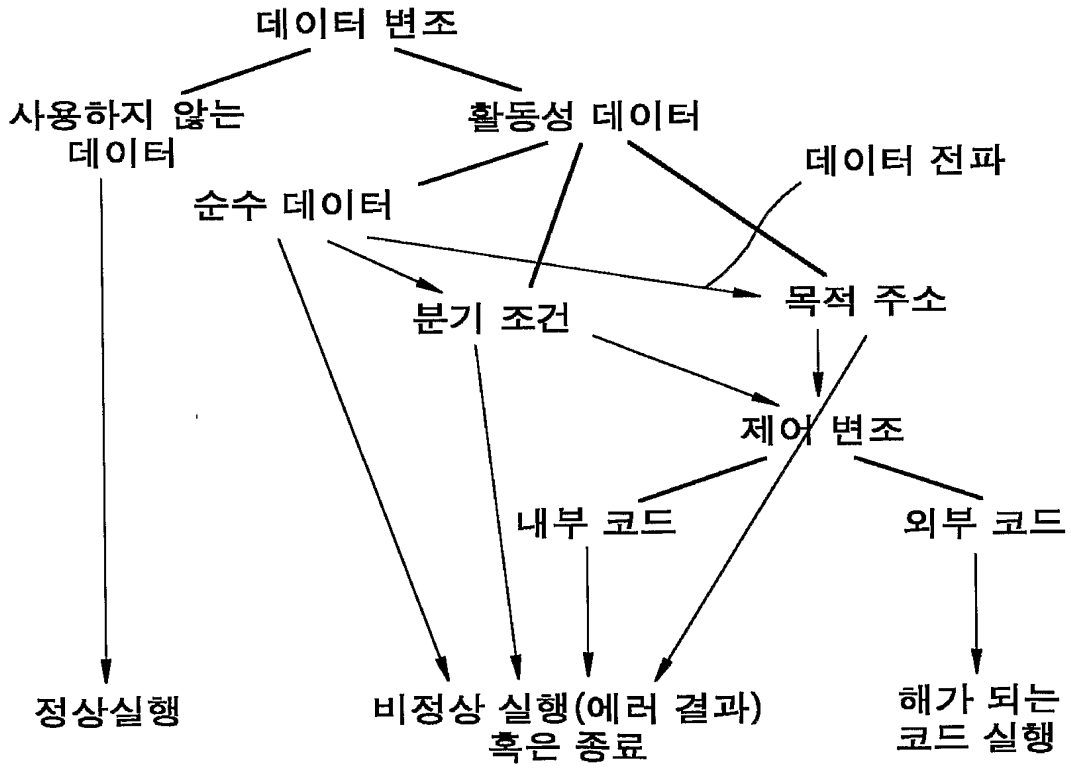
상기 저장관리부가 상기 FIFO 소자에 저장된 내용을 원래의 쓰기 동작들이 기록될 메모리 영역에 저장하는 소정의 간격은 FIFO 소자의 용량으로 정해질 수 있는 것을 특징으로 하는 버퍼 오버플로우 공격으로부터 복구 장치.

【도면】

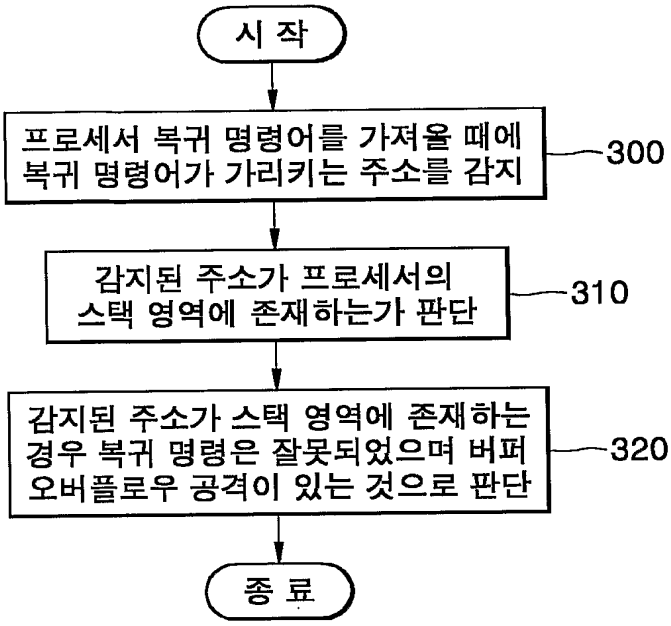
【도 1】


상위 어드레스
하위 어드레스
다이나믹 데이터 세그먼트
스태틱 데이터 세그먼트
텍스트 세그먼트
스택 세그먼트
100
120
인자
130
복귀 주소
150
변경된 복귀 주소
140
이전 프레임 포인터
EBP(Frame Pointer)
110
지역변수들
Local Stack Frame
Undefined
ESP(Stack Pointer)

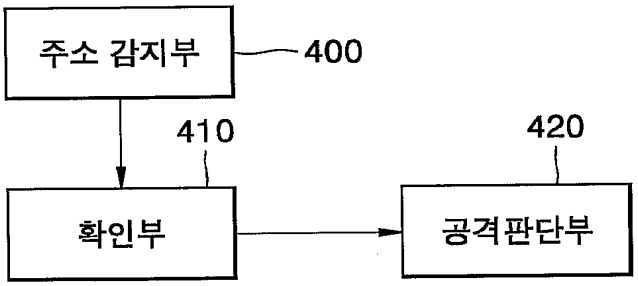
【도 2】


데이터 변조
사용하지 않는 데이터
활동성 데이터
데이터 전파
순수 데이터
분기 조건
목적 주소
제어 변조
내부 코드
외부 코드
정상실행
비정상 실행(에러 결과) 혹은 종료
해가 되는 코드 실행

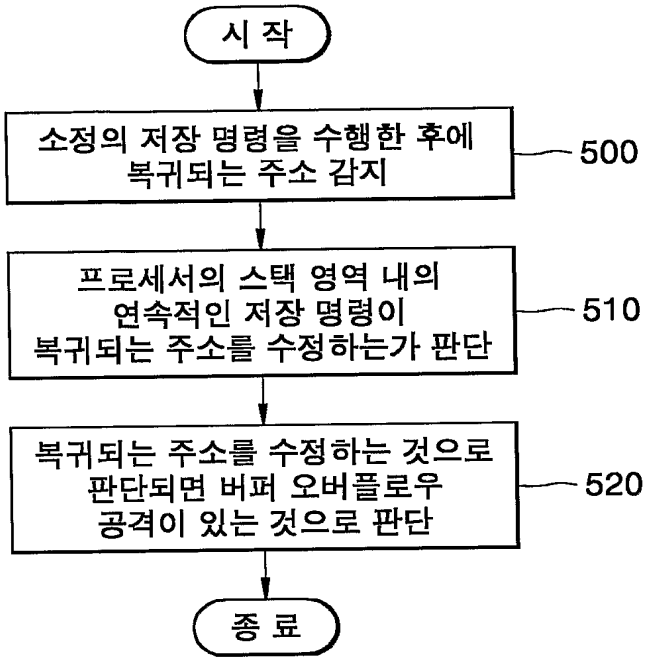
【도 3】


시 작
프로세서 복귀 명령어를 가져올 때에 복귀 명령어가 가리키는 주소를 감지
300
감지된 주소가 프로세서의 스택 영역에 존재하는가 판단
310
감지된 주소가 스택 영역에 존재하는 경우 복귀 명령은 잘못되었으며 버퍼 오버플로우 공격이 있는 것으로 판단
320
종 료

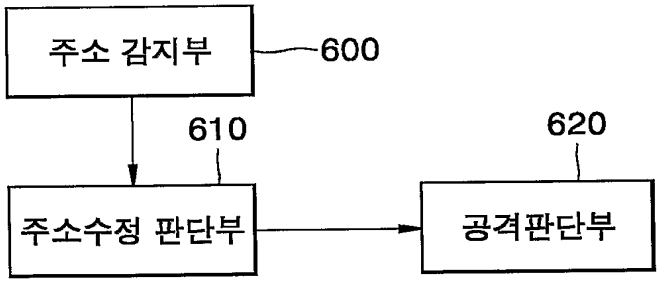
【도 4】


주소 감지부
400
410
420
확인부
공격판단부

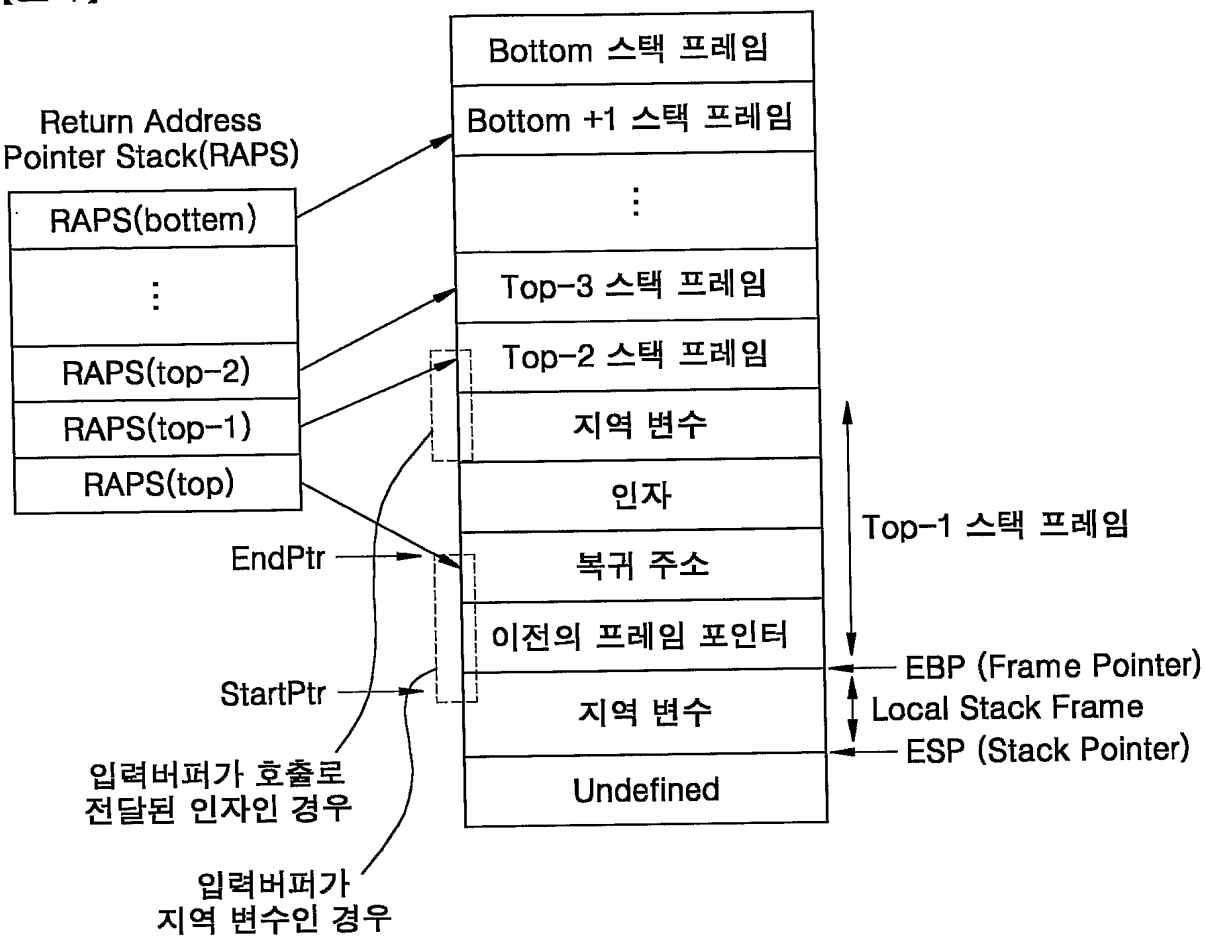
【도 5】


시 작
소정의 저장 명령을 수행한 후에 복귀되는 주소 감지
500
프로세서의 스택 영역 내의 연속적인 저장 명령이 복귀되는 주소를 수정하는가 판단
510
복귀되는 주소를 수정하는 것으로 판단되면 버퍼 오버플로우 공격이 있는 것으로 판단
520
종 료

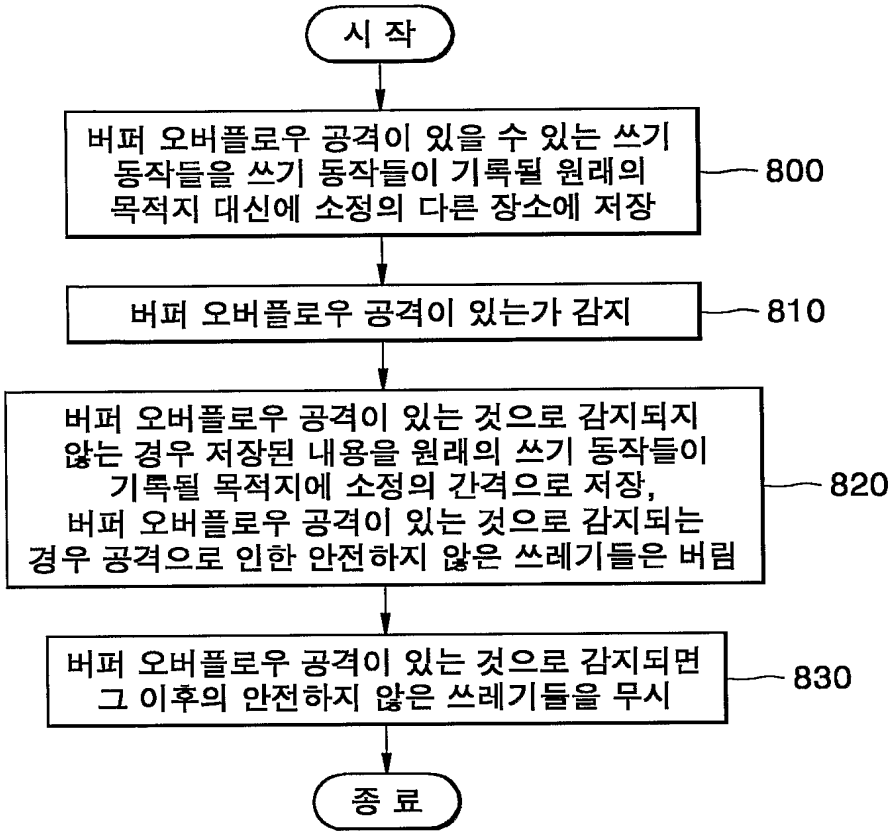
【도 6】


주소 감지부
600
610
620
주소수정 판단부
공격판단부

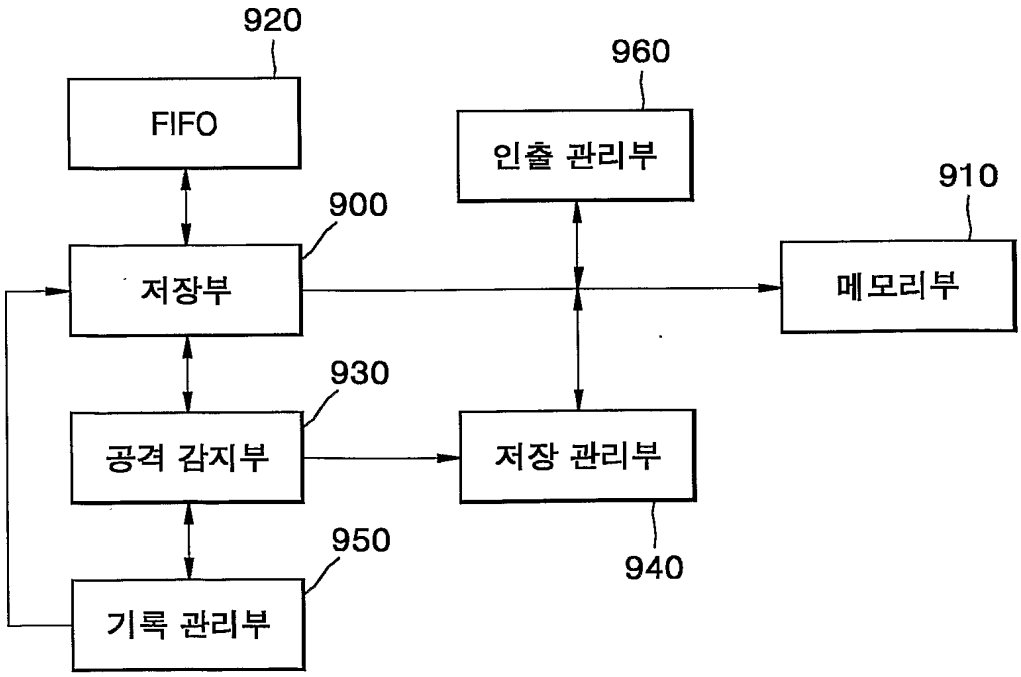
【도 7】


Return Address
Pointer Stack(RAPS)
RAPS(bottem)
RAPS(top-2)
RAPS(top-1)
RAPS(top)
EndPtr
StartPtr
입력버퍼가 호출로
전달된 인자인 경우
입력버퍼가
지역 변수인 경우
Bottom 스택 프레임
Bottom +1 스택 프레임
Top-3 스택 프레임
Top-2 스택 프레임
지역 변수
인자
복귀 주소
이전의 프레임 포인터
지역 변수
Undefined
Top-1 스택 프레임
EBP (Frame Pointer)
Local Stack Frame
ESP (Stack Pointer)

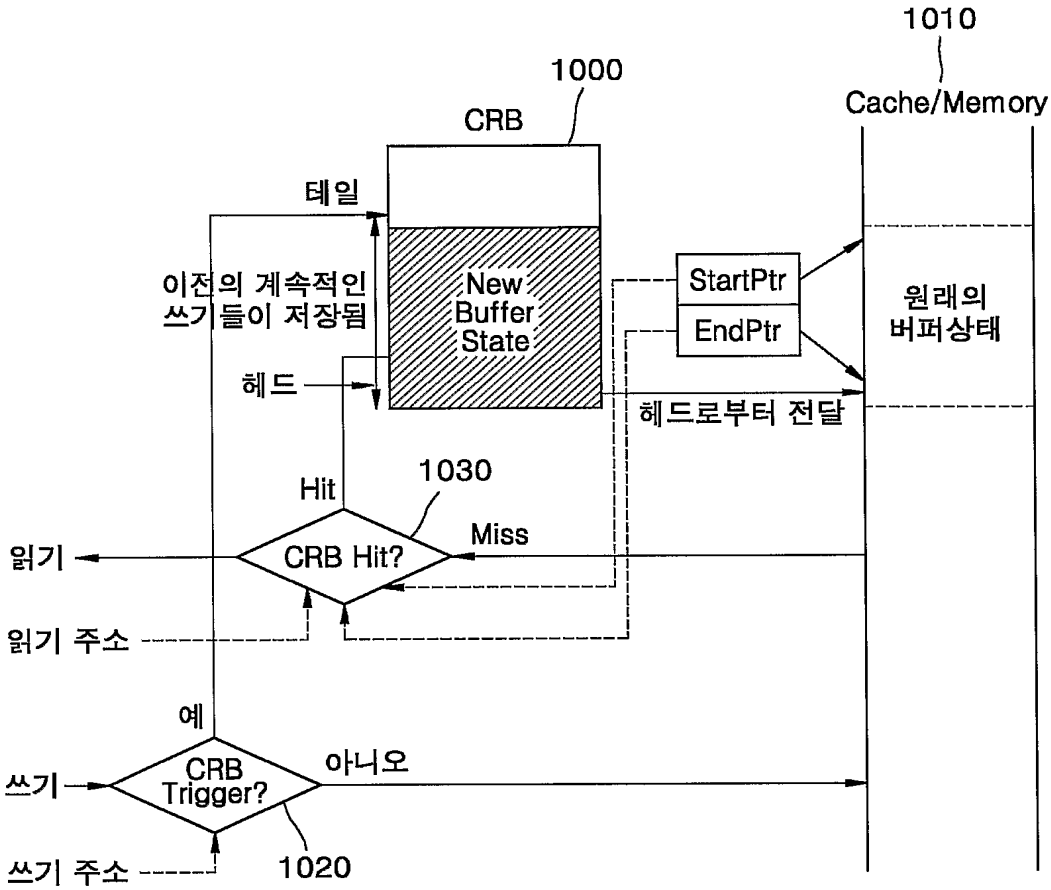
【도 8】


시 작
버퍼 오버플로우 공격이 있을 수 있는 쓰기 동작들을 쓰기 동작들이 기록될 원래의 목적지 대신에 소정의 다른 장소에 저장
800
버퍼 오버플로우 공격이 있는가 감지
810
버퍼 오버플로우 공격이 있는 것으로 감지되지 않는 경우 저장된 내용을 원래의 쓰기 동작들이 기록될 목적지에 소정의 간격으로 저장, 버퍼 오버플로우 공격이 있는 것으로 감지되는 경우 공격으로 인한 안전하지 않은 쓰레기들은 버림
820
버퍼 오버플로우 공격이 있는 것으로 감지되면 그 이후의 안전하지 않은 쓰레기들을 무시
830
종 료

【도 9】


920
FIFO
960
인출 관리부
910
900
저장부
메모리부
930
공격 감지부
저장 관리부
940
950
기록 관리부

【도 10】


1010
Cache/Memory
1000
CRB
테일
이전의 계속적인
쓰기들이 저장됨
New
Buffer
State
StartPtr
EndPtr
원래의
버퍼상태
헤드
헤드로부터 전달
1030
Hit
Miss
읽기
CRB Hit?
읽기 주소
예
쓰기
CRB
Trigger?
아니오
쓰기 주소
1020